

Amendments to the Claims:

This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

1. (Currently amended) A method in a data processing system that implements an object-oriented software environment for translating method calls to version-specific method calls, said method comprising the steps of:
 - providing an interface to an underlying object, applications utilizing said interface to communicate with said underlying object, said interface being separate from said underlying object;
 - generating a plurality of version-specific underlying objects, each one of said version-specific underlying objects being a different version of said underlying object;
 - generating a plurality of translation objects object, each one of said plurality of translation objects for communicating between said interface and [[each]] a different one of said version-specific underlying objects;
 - each one of said plurality of translation objects including a proxy instance and an invocation handler instance;
 - receiving, by a particular one of said plurality of translation objects, a particular interface method call that was invoked on said interface, said particular one of said plurality of translation objects for communicating between said interface and a particular one of said version-specific underlying objects, said particular interface method call including a name and formal parameters;
 - translating, by a particular invocation handler instance that is included in said particular one of said plurality of translation objects, said particular interface method call by determining a version-specific method call that corresponds to said particular interface method call using said name and said formal parameters of said particular interface method call;
 - invoking, by said particular invocation handler instance, said version-specific method call;
 - wherein, said plurality of translation objects are used to translate interface method calls that were invoked on said interface to corresponding version-specific method calls for each version of said underlying object; utilizing said translation object for translating an interface method call invoked on said interface to a version-specific method call for said underlying object for each version of said underlying object; and

generating said plurality of translation objects ~~object~~ from a single proxy class and a single invocation handler class, wherein the same proxy class and invocation handler class are used to generate said plurality of translation objects ~~object for each different version of said underlying object.~~

2. (Currently amended) The method according to claim 1, further comprising the steps of:
determining a current version of said underlying object;
determining whether a ~~[[said]]~~ translation object exists for said current version; ~~[[and]]~~
in response to determining that said translation object does not exist for said current version,
generating said translation object for said current version utilizing a building object~~[[.]]~~; and
including said generated translation object for said current version in said plurality of translation
objects.
3. (Original) The method according to claim 2, further comprising the steps of:
creating said building object utilizing said proxy class and said invocation handler class.
4. (Currently amended) The method according to claim 1, further comprising the steps of:
generating a building instance of said proxy class and said invocation handler class;
utilizing said building instance to generate said proxy instance and said [[an]] invocation handler
instance for each one of said plurality of translation objects, ~~said interface;~~
~~utilizing said building instance to generate said translation object; and~~
~~including said invocation handler instance in said translation object.~~
5. (Canceled)
6. (Canceled)
7. (Currently amended) The method according to claim 1 ~~[[6]]~~, further comprising the steps of:
invoking said particular interface method call ~~on said translation instance~~ by a client object; and
returning, by said particular invocation handler instance, said return object to said client object.
8. (Currently amended) The method according to claim 1 ~~[[5]]~~, further comprising the steps of:
~~utilizing, by said invocation handler instance, said interface method call and original parameters~~
~~included in said interface method call to locate said version specific method call;~~

determining whether any of said formal original parameters are proxy objects; and
in response to determining that at least one of said formal original parameters are proxy objects,
replacing each said proxy object with an underlying object type for said particular one of said version-
specific underlying objects ~~version~~.

9. (Original) The method according to claim 7, further comprising the steps of:
determining whether said return object is a second interface; and
in response to a determination that said return object is said second interface, creating a second
proxy instance that implements a return type for said second interface.

10. (Original) The method according to claim 7, further comprising the steps of:
determining whether said return object is a second interface;
in response to determining that said return object is said second interface, determining whether
said return type is an array; and
in response to determining that said return object is an array, creating a separate proxy instance
that implements a return type for said second interface for each element of said array.

11. (Canceled)

12. (Currently amended) A data processing system that implements an object-oriented software
environment for translating method calls to version-specific method calls, comprising:
an interface to an underlying object, applications utilizing said interface to communicate with said
underlying object, said interface being separate from said underlying object;
a plurality of version-specific underlying objects, each one of said version-specific underlying
objects being a different version of said underlying object;
a plurality of translation objects, each one of said plurality of translation objects for
communicating between said interface and a different one of said version-specific underlying objects;
~~a translation object for communicating between said interface and each one of said version-~~
~~specific underlying objects;~~
each one of said plurality of translation objects including a proxy instance and an invocation
handler instance;
a particular one of said plurality of translation objects receiving a particular interface method call
that was invoked on said interface, said particular one of said plurality of translation objects for

communicating between said interface and a particular one of said version-specific underlying objects,
said particular interface method call including a name and formal parameters;

a particular invocation handler instance that is included in said particular one of said plurality of
translation objects translating said particular interface method call by determining a version-specific
method call that corresponds to said particular interface method call using said name and said formal
parameters of said particular interface method call;

said particular invocation handler instance invoking said version-specific method call;

wherein, said plurality of translation objects are used to translate interface method calls that were
invoked on said interface to corresponding version-specific method calls for each version of said
underlying object; and

said translation object being utilized for translating an interface method call invoked on said
interface to a version-specific method call for said underlying object for each version of said underlying
object; and

said plurality of translation objects translation object being generated from a single proxy class
and a single invocation handler class, wherein the same proxy class and invocation handler class are used
to generate said plurality of translation objects translation object for each different version of said
underlying object.

13. (Currently amended) The system according to claim 12, further comprising:

said data processing system including a CPU executing code for determining a current version of
said underlying object;

said CPU executing code for determining whether a [[said]] translation object exists for said
current version; [[and]]

in response to determining that said translation object does not exist for said current version, said
translation object for said current version being generated utilizing a building object[[.]]; and

said generated translation object for said current version being included in said plurality of
translation objects.

14. (Original) The system according to claim 13, further comprising:

said building object being created utilizing said proxy class and said invocation handler class.

15. (Currently amended) The system according to claim 12, further comprising:

a building instance of said proxy class and said invocation handler class;

said building instance being utilized to generate said proxy instance and said [[an]] invocation handler instance for said interface for each one of said plurality of translation objects;

~~said building instance being utilized to generate said translation object; and including said invocation handler instance in said translation object.~~

16. (Canceled)

17. (Canceled)

18. (Currently amended) The system according to claim 12 [[17]], further comprising:
said particular interface method call being invoked ~~on said translation instance~~ by a client object;
and
said particular invocation handler instance returning said return object to said client object.

19. (Original) The system according to claim 12 [[16]], further comprising:
~~said invocation handler instance utilizing said interface method call and original parameters included in said interface method call to locate said version specific method call;~~
said CPU executing code for determining whether any of said formal ~~original~~ parameters are proxy objects; and
in response to determining that at least one of said ~~original~~ formal parameters are proxy objects, said CPU executing code for replacing each said proxy object with an underlying object type for said particular one of said version-specific underlying objects ~~version~~.

20. (Original) The system according to claim 18, further comprising:
said CPU executing code for determining whether said return object is a second interface; and
in response to a determination that said return object is said second interface, said CPU executing code for creating a second proxy instance that implements a return type for said second interface.

21. (Original) The system according to claim 18, further comprising:
said CPU executing code for determining whether said return object is a second interface;
in response to determining that said return object is said second interface, said CPU executing code for determining whether said return type is an array; and

in response to determining that said return object is an array, said CPU executing code for creating a separate proxy instance that implements a return type for said second interface for each element of said array.

22. (Canceled)

23. (Currently amended) A computer program product, recorded on a computer-readable medium, in a data processing system that implements an object-oriented software environment for translating method calls to version-specific method calls, said product comprising:

instruction means for providing an interface to an underlying object, applications utilizing said interface to communicate with said underlying object, said interface being separate from said underlying object;

instruction means for generating a plurality of version-specific underlying objects, each one of said version-specific underlying objects being a different version of said underlying object;

instruction means for generating a plurality of translation objects, ~~translation object~~ each one of said plurality of translation objects for communicating between said interface and a different [[each]] one of said version-specific underlying objects;

each one of said plurality of translation objects including a proxy instance and an invocation handler instance;

instruction means for receiving, by a particular one of said plurality of translation objects, a particular interface method call that was invoked on said interface, said particular one of said plurality of translation objects for communicating between said interface and a particular one of said version-specific underlying objects, said particular interface method call including a name and formal parameters;

instruction means for translating, by a particular invocation handler instance that is included in said particular one of said plurality of translation objects, said particular interface method call by determining a version-specific method call that corresponds to said particular interface method call using said name and said formal parameters of said particular interface method call;

instruction means for invoking, by said particular invocation handler instance, said version-specific method call;

wherein, said plurality of translation objects are used to translate interface method calls that were invoked on said interface to corresponding version-specific method calls for each version of said underlying object; and

~~instruction means for utilizing said translation object for translating an interface method call invoked on said interface to a version specific method call for said underlying object for each version of said underlying object; and~~

instruction means for generating said plurality of translation objects ~~translation object~~ from a single proxy class and a single invocation handler class, wherein the same proxy class and invocation handler class are used to generate said plurality of translation objects ~~translation object for each different version of said underlying object.~~

24. (Currently amended) The product according to claim 23, further comprising:

instruction means for determining a current version of said underlying object;

instruction means for determining whether a [[said]] translation object exists for said current version; [[and]]

in response to determining that said translation object does not exist for said current version, instruction means for generating said translation object for said current version utilizing a building object~~[[.]]~~; and

instruction means for including said generated translation object for said current version in said plurality of translation objects.

25. (Original) The product according to claim 24, further comprising:

instruction means for creating said building object utilizing said proxy class and said invocation handler class.

26. (Currently amended) The product according to claim 23, further comprising:

instruction means for generating a building instance of said proxy class and said invocation handler class;

instruction means for utilizing said building instance to generate said proxy instance and said [[an]] invocation handler instance ~~for said interface;~~

~~instruction means for utilizing said building instance to generate said translation object; and instruction means for including said invocation handler instance in said translation object.~~

27. (Canceled)

28. (Canceled)

29. (Currently amended) The product according to claim 23 [[28]], further comprising:
instruction means for invoking said particular interface method call ~~on said translation instance~~
by a client object; and
instruction means for returning, by said particular invocation handler instance, said return object
to said client object.
30. (Currently amended) The product according to claim 23 [[27]], further comprising:
~~instruction means for utilizing, by said invocation handler instance, said interface method call and~~
~~original parameters included in said interface method call to locate said version-specific method call;~~
instruction means for determining whether any of said ~~original~~ formal parameters are proxy
objects; and
in response to determining that at least one of said ~~original~~ formal parameters are proxy objects,
instruction means for replacing each said proxy object with an underlying object type for said particular
one of said version-specific underlying objects ~~version~~.
31. (Original) The product according to claim 29, further comprising:
instruction means for determining whether said return object is a second interface; and
in response to a determination that said return object is said second interface, instruction means
for creating a second proxy instance that implements a return type for said second interface.
32. (Original) The product according to claim 29, further comprising:
instruction means for determining whether said return object is a second interface;
in response to determining that said return object is said second interface, instruction means for
determining whether said return type is an array; and
in response to determining that said return object is an array, instruction means for creating a
separate proxy instance that implements a return type for said second interface for each element of said
array.
33. (Canceled)